PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-126161
(43)Date of publication of application : 11.05.1999

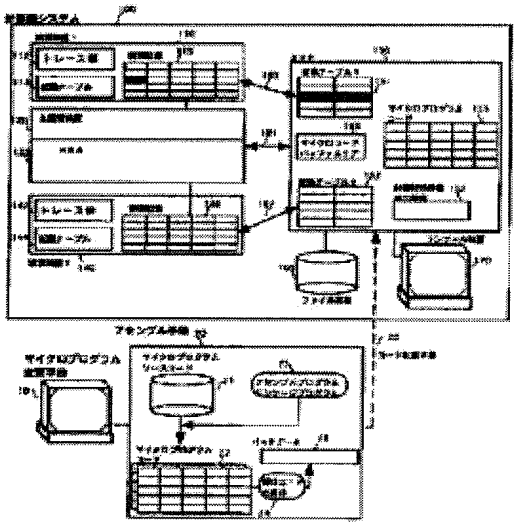
(51)Int.Cl. G06F 9/22
G06F 9/22

(21)Application number : 09-290758
(22)Date of filing : 23.10.1997

(71)Applicant : HITACHI LTD
HITACHI INFORMATION TECHNOLOGY CO LTD
(72)Inventor : TAKESHIMA SEISUKE

(54) FAULT AVOIDING METHOD FOR CONTROL MEMORY

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a fault avoiding method for control memory that facilitates the maintenance management of a microprogram.
SOLUTION: The correspondence of an address (physical address) on a control memory 115 and an address (logical address) owned by the source code of the microprogram is added to a microprogram code 22 for loading to the control memory 115. At the time of initial microprogram load, an SVP tests the control memory and generates the correspondence table (translation table) of the physical address and the logical address so as to avoid the defective part of that memory, and based on the correspondence table, the physical address of the microprogram code 22 is rearranged so as to execute the initial load. The generated correspondence table is maintained as it is.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平11－126161

(43)公開日　平成11年(1999)５月11日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　9/22	３８０	Ｇ０６Ｆ　9/22	３８０Ｂ
	３１０		３１０Ｄ

審査請求　未請求　請求項の数４　ＯＬ　（全 18 頁）

(21)出願番号　特願平9－290758

(22)出願日　平成９年(1997)10月23日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(71)出願人　000153454
株式会社日立インフォメーションテクノロジー
神奈川県秦野市堀山下１番地

(72)発明者　竹島　靖祐
神奈川県秦野市堀山下１番地　株式会社日立インフォメーションテクノロジー内

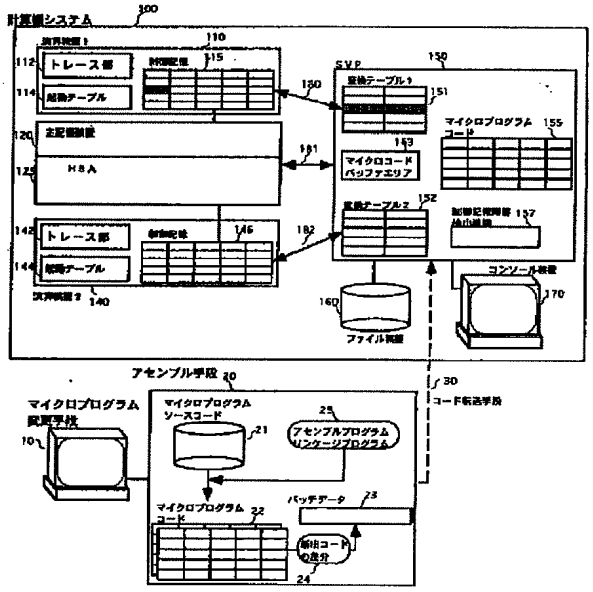
(74)代理人　弁理士　鈴木　誠

(54)【発明の名称】　制御記憶の障害回避方法

(57)【要約】

【課題】　マイクロプログラムの保守管理を容易にする制御記憶の障害回避方法を提供する。

【解決手段】　制御記憶にロードするためのマイクロプログラムコードに、制御記憶上のアドレス（物理アドレス）とマイクロプログラムのソースコードが持つアドレス（論理アドレス）の対応を付加する。ＳＶＰは、マイクロプログラム初期ロード時、制御記憶をテストし、その不良部位を回避するように物理アドレスと論理アドレスの対応表（変換テーブル）を生成し、該対応表にもとづいて、マイクロプログラムコードの物理アドレスを再配置し、初期ロードを実行する。生成した対応表はそのまま保守しておく。



1

【特許請求の範囲】

【請求項１】　マイクロプログラムが格納された制御記憶を内蔵する一つあるいは複数の演算装置と、前記演算装置の保守を司る保守制御装置とを具備してなる計算機システムにおいて、

制御記憶にロードするためのマイクロプログラムコードに、制御記憶上のアドレス（以下、物理アドレスと称す）とマイクロプログラムのソースコードが持つアドレス（以下、論理アドレスと称す）の対応を付加して保持し、

保守制御装置は、マイクロプログラムを制御記憶に初期ロードする際、制御記憶をテストし、その不良部位を回避するように物理アドレスと論理アドレスとの対応表を生成し、該対応表に基づいて、前記マイクロプログラムソースコードに付加された物理アドレスを再配置し、該再配置後の物理アドレスに従ってマイクロプログラムを制御記憶にロードすることを特徴とする制御記憶の障害回避方法。

【請求項２】　請求項１記載の制御記憶の回避方法において、保守制御装置は、計算機システム稼働中の制御記憶障害発生時、その不良部分を回避するように対応表中の物理アドレスと論理アドレスの対応を再配置し、該配置された論理アドレスと物理アドレスに基づいて、該当マイクロプログラムコードに付加された物理アドレスを再配置し、該再配置後の物理アドレスに従って当該マイクロプログラムコードを制御記憶に再ロードすることを特徴とする制御記憶の障害回避方法。

【請求項３】　請求項１、２記載の制御記憶の障害回避方法において、保守制御装置は、物理アドレスの再配置時、対応表の物理アドレスにより、マイクロプログラムの起動アドレスを登録したテーブルの該当エントリを書き換えることを特徴とする制御記憶の障害回避方法。

【請求項４】　請求項１、２記載の制御記憶の障害回避方法において、保守制御装置は、実行状況のトレース時、トレース情報のマイクロプログラムの物理アドレスを、対応表により論理アドレスに変換することを特徴とする制御記憶の障害回避方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、マイクロプログラム制御の計算機システムにおける制御記憶の障害部位を回避する方法に関する。

【０００２】

【従来の技術】現在マイクロプログラムにより制御される演算装置をもつ計算機システムでは、マイクロプログラムによる計算機システムの障害処理等の事情でマイクロプログラムの量、即ち制御記憶の容量は増大の傾向にあり、それだけ制御記憶自身の耐障害機能が求められている。

【０００３】従来、制御記憶内の固定障害の発生に対して、計算機システムに接続された保守・診断専用のサービスプロセッサ（ＳＶＰ）に内蔵されたファイル機構、あるいは主記憶装置上のハードウェア専用領域（ＨＳＡ）よりマイクロプログラムを再ロードする時、制御記憶の空白（リザーブ）領域に障害発生部位のアドレスを持つマイクロプログラムをロードし、その後の演算装置の動作時に、マイクロプログラムをデコードする機構により制御記憶の障害発生部位アドレスをリザーブ領域のアドレスをに変換する、又はマイクロプログラムのアドレスを通常の計算機命令語の様に基底アドレスレジスタを使用した相対アドレスによる再配置可能な構造にし、基底アドレスレジスタを変更する、等により障害発生部の使用を回避する方法が一般に知られている。しかし、これらの方法では、マイクロプログラム実行時、必ずアドレス変換、あるいは基底アドレス加算のプロセスが必要となり、演算装置の性能が低下し、演算装置の構造が複雑となり、計算機システムのコストアップを招く欠点がある。

【０００４】一方、これらの欠点を回避するため、次のような機能により制御記憶の固定障害を抑止する方法がある。

１、制御記憶内の固定障害を検出すると演算装置の実行を一時中断し、その事を障害の起きたアドレス情報とともにＳＶＰに通報する。

２、ＳＶＰは自身が持つマイクロプログラムのアドレス管理情報を使用して、障害の起きたアドレスを持つマイクロプログラムのワードをＳＶＰが持つファイル装置内のマイクロプログラムソースファイルから検索し、そのワードを制御記憶内の空いた部分にアサインしてアドレスを付け替えるため、ソースファイルを書き換える。

３、その後、ＳＶＰはソースファイルをアセンブル（リンケージ）し、新たなマイクロプログラムのコード列を生成する。このコード列は障害の起きたアドレス部位はアサインされないように空白となっているため、制御記憶に再ロードしてもその影響を被ることはない。また、ＳＶＰはアドレス管理情報に制御記憶の障害の部分が使用不可であることを記録する。

４、新たなマイクロプログラムコードをロードし、その後演算装置の動作を再開する。

【０００５】なお、このような制御記憶の固定障害回避手法は、例えば特開平６－８３６１３号公報や特開平６－１８７１４７号公報などに記載されている。

【０００６】

【発明が解決しようとする課題】上記方法は、演算装置に制御記憶の障害発生部位の使用を回避するための特別な機構を設ける必要がない利点があるが、以下に記すような多くの問題点があり、現実の計算機システムに適用するのは困難である

１、計算機システムのマイクロプログラムを保守する為、制御記憶上のマイクロプログラムのあるアドレスの

ワードにパッチを適用する場合、前記アドレス移動が行われている可能性がある為、パッチを適用することが不可能である。又、この様な計算機システムに新しいマイクロプログラムをインストールする場合、それが障害発生部位を使用することになると、システムダウンを起こす事が考えられる。これらを回避するためには、マイクロプログラム改訂対象の計算機システムのＳＶＰが持つマイクロプログラムソースファイル及びアドレス管理情報を取り寄せ、そのソースファイルを管理情報に従い障害発生部を回避する様に改訂し、当該計算機システムのＳＶＰに送り、前記した手順により再アセンブルしロードすることになるが、この様な手順が顧客サイトにある計算機システム毎に必要となり、従来の様な全ての計算機システムに対し共通のパッチ、又は新マイクロプログラムのインストールにより変更を施す場合と比べてマイクロプログラム保守が非常に煩雑になる。

【０００７】２、マイクロプログラム制御の演算装置では、命令、割込等の処理は制御記憶の特定の固定番地から始まるマイクロプログラムにより処理を実行するのが一般的であるが、その固定番地が前記再配置により移動すると、そのマイクロプログラムは動作不可能となり、演算装置は処理を続行できなくなる。

【０００８】３、計算機システムで障害解析のために、マイクロプログラムの実行状況を知りたい場合、ハードウエアのトレース機能によりマイクロプログラムのトレースを採取し解析したい場合がある。このとき前記アドレス移動が行われていると、その移動したマイクロプログラムのトレース情報は、マイクロプログラム設計者が持つ初期ソースファイル（計算機システム出荷時にコードを生成したソースファイル、即ち制御記憶障害による移動前の古いアドレスを持つ）による参照／照合が出来ないため、解析不可能であり、トレースを解析するためには、その計算機システムのＳＶＰが持つソースファイル及びアドレス管理情報を取り寄せなければならなくなり、保守／デバッグ手順は煩雑である。

【０００９】４、計算機システムが演算装置を複数個持ち、それらが共通のＳＶＰにより制御及び保守されているマルチプロセッサシステムである場合、特に各演算装置上の制御記憶が共通のマイクロプログラムソースファイルにより管理されている場合に、ある一個の演算装置の制御記憶障害で、ＳＶＰ内のソースファイルに前記アドレス移動が適用された後、システム内の別演算装置の制御記憶にそのマイクロプログラムがロードされると、最初の演算装置の制御記憶の障害発生アドレスと同じアドレス部位は使用不可能にされてしまう、即ち計算機システム内の全演算装置の制御記憶に各演算処理装置個別に発生する制御記憶障害の総和分の使用不可能な領域が出来てしまう事になる。この様な制御記憶の無効領域の増大は、アドレス移動が不可能となったり、又、機能追加等の為にマイクロプログラムの更新（追加）に対応ができなくなる可能性がある。また、この様な無効領域を見越して制御記憶を増やすことは、計算機システムのコストアップをもたらす事も考えられる。

【００１０】自明な回避策として、各演算装置に対応してソースファイルを持ち、制御記憶障害によるアドレス移動は各々のソースファイルを基に遂行し、他の演算装置に影響を及ぼす事を避けることが考えられるが、この解決策では前記１で述べたのと同様の問題点により、各演算処理装置毎にマイクロプログラム管理を行わなければならなくなり、マイクロプログラム保守は更に煩雑になる。

【００１１】５、計算機システム内で、アドレス移動に必要な制御記憶の空き領域が存在しない場合、制御記憶に障害が起こるとシステムダウンとなるが、事前にそれを知る手段を持たないため、保守が不可能である。

【００１２】本発明の目的は、マイクロプログラム制御の計算機システムにおいて、制御記憶の障害部位を回避する際の上記のような問題点を解決し、マイクロプログラムの保守操作性の向上を図ることにある。

【００１３】

【課題を解決するための手段】本発明は、制御記憶にロードするために、ＳＶＰ上のファイルに置かれるマイクロプログラムコードに、制御記憶上のアドレス（物理アドレス）とマイクロプログラムのソースコードが持つアドレス（論理アドレス）の対応を付加する。

【００１４】マイクロプログラムを制御記憶に初期ロードする際、ＳＶＰは、制御記憶をテストし、その不良部位を回避するように物理アドレスと論理アドレスとの対応表を生成し、該対応表に基づいて、前記マイクロプログラムソースコードに付加された物理アドレスを再配置し、該再配置後の物理アドレスに従ってマイクロプログラムを制御記憶にロードする生成した対応表は、計算機システム起動後も、そのまま保持しておく。

【００１５】計算機システム稼働中の制御記憶障害発生時、ＳＶＰは、その不良部分を回避するように対応表中の物理アドレスと論理アドレスの対応を再配置し、該配置された論理アドレスと物理アドレスに基づいて、該当マイクロプログラムコードに付加された物理アドレスを再配置し、該再配置後の物理アドレスに従って当該マイクロプログラムコードを制御記憶に再ロードする。

【００１６】また、物理アドレスの再配置時、ＳＶＰは、対応表の物理アドレスにより、マイクロプログラムの起動アドレスを登録したテーブルの該当エントリを書き換える。

【００１７】さらに、計算機システム実行状況のトレース時ＳＶＰは、トレース情報のマイクロプログラムの物理アドレスを、対応表により論理アドレスに変換して出力する。

【００１８】なお、マイクロプログラムコードを生成するアセンブル手段（またはリンクエディット手段）にお

いて、マイクロプログラムコード及びパッチ作成時にマイクロプログラムコードとパッチに論理アドレスを付加するようにする。これにより、顧客サイト毎に存在する計算機システムに対して共通のマイクロプログラムコード及びパッチを適用することができる。

【００１９】また、マルチプロセッサシステムである場合、各演算装置対応にアドレス対応表を持たせることにより、一個の演算装置で発生した制御記憶の障害をその演算装置固有のアドレス対応表により、その演算装置のみのマイクロプログラムの再配置を行うことで、他の演算装置の制御記憶の同一アドレス部位が使用できなくなることを防ぎ、制御記憶の無効領域が生じないようにできる。

【００２０】さらには、制御記憶の障害による、マイクロプログラムの再配置の履歴をＳＶＰのファイルに記録し、保守操作員の入力するコマンドによりＳＶＰのコンソール装置にその履歴を表示する機構、あるいはＳＶＰがその履歴及び前記アドレス対応表を監視し、制御記憶の空き領域が十分でない場合、その事を自動的に警告メッセージとして表示するようにする。

【００２１】

【発明の実施の形態】以下、本発明の実施の形態について図面により具体的に説明する。

【００２２】図１は本発明に係わる一実施の形態の全体構成図である。計算機システム１００は演算装置１、それに接続された主記憶装置１２０、及び、これらの装置の保守を司る保守制御装置であるＳＶＰ１５０により構成されている。マルチプロセッサシステムの場合、演算装置２が構成に含まれる場合もある。なお、図１では演算装置が３個以上の場合は図示していないが、同様の演算装置が主記憶装置に接続可能である。演算装置１には、マイクロプログラムを格納する制御記憶１１５、演算装置１の実行状況をトレースするトレース部１１２、マイクロプログラムの起動アドレスを登録した起動テーブル１１４をそれぞれ含む。演算装置２の構成も同様である。主記憶装置１２０上には、ハードウエア専用領域であるＨＳＡ１２５が存在し、ＨＳＡ１８１とＳＶＰ１５０間にはデータ転送パス１８１を具備する。又、制御記憶１１５、１４６へのアクセスパス１８０、１８２により、演算装置１、２あるいはＳＶＰ１５０による制御で制御記憶１１５、１４６にデータを転送する事が出来る。ＳＶＰ１５０は、それ自身が持つ記憶装置にはマイクロプログラムコード１５５、各演算装置１、２に対応したアドレス変換テーブル（物理アドレスと論理アドレスの対応表）１５１、１５２が有り、さらに、制御記憶１１５、１４６にマイクロプログラムコードを転送する一時記憶であるバッファエリア１５３、アクセスパス１８０、１８２を使用し制御記憶１１５、１４６の障害を検出する制御記憶障害検出機構１５７を具備する。又、ＳＶＰ１５０上のテーブルのバツクアップを保持するファイル装置１６０、ＳＶＰ１５０を通じて計算機システム１００を操作するためのコンソール装置１７０が接続されている。

【００２３】なお、ＳＶＰ１５０の記憶装置上にある変換テーブル１５１、１５２、マイクロプロラムコード５５、及びバッファエリア１５３は、主記憶装置１２０のＨＳＡ１２５上に置く事もできる。この場合、演算装置により変換テーブルの更新を行う事が可能でる。ただし、これはマルチプロセッサシステムで１個の演算装置の障害発生によるテーブル操作を他の演算装置で実行する場合に限られる。

【００２４】以上の計算機システム１００は顧客サイトに設置されるが、以下の設備はマイクロプログラム設計者側に１システム有ればよい。

【００２５】アセンブル手段２０はそれ自体計算機であり、それが内蔵するファイル装置にはマイクロプログラムソースコード２１を持っている。マイクロプログラム設計者は、該アセンブル手段２０に接続されたコンソール装置であるマイクロプログラム変更手段１０を用いてソースコードを変更した場合、アセンブル手段２０に装備されたアセンブルプログラム及びリンケージプログラム２５によりマイクロプログラムコード２２を作成することができ、又、論理処理部２４により新旧のコード２２を比較しパッチデータ２３を作成できる。マイクロプログラム設計者は、これらのマイクロプログラムコード２２、パッチデータ２３をコード転送手段３０により顧客サイトに設置された各計算機システムに送る事ができる。なお、コード転送手段３０はローカルエリアネットワーク、公衆回線による転送、あるいはフロッピーディスク、磁気テープ等による媒体の搬送の何れかにより実現される。

【００２６】図２は、前記ＳＶＰ１５０上のマイクロプログラムコード１５５、アドレス変換テーブル１５１、１５２の構成例を詳細に記述したものである。

【００２７】変換テーブル２００はエントリが上から制御記憶の物理アドレス０より始まる順列を構成し、各エントリには、有効フラグ２０１、マイクロプログラムソースコードが持つ論理アドレス２０２からなり、これにより物理アドレスから論理アドレスの変換を行う事が出来る。有効フラグ２０１はその物理アドレスが使用されているかを表し、制御記憶の空き領域を検索するために使用される。前述したように、この変換テーブルは各演算装置に対応して１個存在する。例えば２００は図１の演算装置１１０のアドレス変換テーブル１５１に対応し、２２０は別の演算装置１４０のアドレス変換テーブル１５２に対応する。

【００２８】マイクロプログラムコード３００は図１の同マイクロプログラムコード１５５に対応し、マイクロコードテーブル３１０、クロスアァレンステーブル３３０よりなる。マイクロコードテーブル３１０は、各１行

のエントリがマイクロプログラムの１ワードに対応し、上からマイクロプログラムソースコードの論理アドレス０より始まる順列を構成する。各エントリはそれぞれ、物理アドレス３１３、分岐先論理アドレス３１４、オブジェクトコード３１５、フラグ３１６、及びクロスリファレンステーブル３３０を指すポインタ３１７にフィールド分割されている。物理アドレス３１３には、３４０で示すに各演算装置に対応した物理アドレスを登録できる。同様にオブジェクトコード３１５の一部である分岐アドレス（論理アドレス３１４で指される分岐先ワードの物理アドレス）も、３５０で示す様に、各演算装置に対応した物理アドレスを持つことが出来る。フラグ３１６はアドレス再配置時に使用される。クロスリファレンステーブル３３０は分岐による参照を遡るために使用されるテーブルである。例えば、ポインタ３１７が指し示すエントリ３３２は、そのワードに分岐するワードの論理アドレスであり、次のポインタ３３４はそのワードに分岐する別のワードが存在した場合、それらワードのポインタを該テーブル３３０上に登録しチェイニングする為のものである。

【００２９】図３に、テーブル３１０、３３０上のマイクロプログラムワード間の相互関係を示す。今、マイクロプログラムソースコード上でワードA、及びワードBからワードCへ分岐しているとする。この場合、テーブル３１０のワードCのエントリ上のポインタ３１７が指すテーブル３３０のエントリ３３２には、テーブル３１０上のワードAのエントリの論理アドレスが登録され、更にポインタ３３４が指すエントリにはテーブル３１３上のワードBの論理アドレスが登録される。図３の例では、これ以上ワードCに分岐するワードが無いため、ワードBの論理アドレスが登録されるエントリ３３２に対応するポインタ３３４には０が登録される。これらのテーブル３１０、３３０は、初期状態でのテーブル間の参照関係がアセンブル（及びリンケージ）時に決定される。クロスリファレンステーブル３３０は、あるワードのアドレスが変更されたとき、影響を受ける他のワードを特定するために使用される。これについては後述する。

【００３０】図２に戻り、アセンブル手段（図１の２０）によりマイクロプログラムコードが作成された時点では、マイクロプログラムコード３００一式が生成され、転送手段３０により計算機システム１００のＳＶＰ１５０に転送される。その時のアドレス対応は論理アドレス＝物理アドレスで各エントリが登録され、特に図２の３４０、３５０の物理アドレスは全て論理アドレスが初期値として設定される。これが、後述するように、アドレス再配置により対応する物理アドレスに書き換えられる。

【００３１】パッチデータ５００は、計算機システム１００のマイクロプログラムアップデート時にアセンブル手段２０より送られるマイクロコードデータの１ワード分のフォーマットであり、必要に応じたワード数のデータをＳＶＰ１５０が受け取れる。パッチデータ５００は、アップデート対象のワードの論理アドレス５０２、そのワードから分岐する先のワードを示す論理アドレス５０３、マイクロコード本体のオブジェクトコード５０４、このデータの属性を示すフラグ５０５からなり、フラグ５０５はそのワードが追加／削除／上書きのうち何れかであることを示す。

【００３２】マイクロコードバッファエリア６００は、図１の同バッファエリア１５３の構造を詳細に記述したものである。先にも述べたように、マイクロコードバッファエリア６００はアドレス再配置又はマイクロプログラムのアップデート時に一時的に使用されるエリアであり、ＳＶＰ１５０或いは主記憶装置１２０上のＨＳＡ１２５に置かれて制御記憶１１５、１４６にロードされるべきデータ等をバッファリングするもので、アップデートするマイクロプログラムのワードの物理アドレス６０１、そのエントリをロードするか否かを示す有効フラグ６０２、ロードするオブジェクトコード６０３からなる。なお、オブジェクトコードは分岐アドレスを含むが、後に説明するようにアドレス再配置による物理アドレスの変更は解決された状態でロードされる。

【００３３】図４は、図１の計算機システムにおけるマイクロプログラム初期ロードのフローチャートである。以下、図２および図４を参照してマイクロプログラム初期ロード時の動作を説明する。

【００３４】マイクロプログラム初期ロードは、計算機システム１００を起動する時、ＳＶＰ１５０のマイクロプログラムコード１５５（図２の３００）を、各演算装置１１０、１４０内の制御記憶１１５、１４６にロードすることで行われる。この時点では、マイクロプログラムコードには再配置は行われておらず、アセンブル手段２０によって設定された論理アドレス＝物理アドレスの状態で、図２の変換テーブル２００、２２０及びマイクロプログラムコード３００が存在する。

【００３５】マイクロプログラム初期ロードは、ＳＶＰ１５０が演算装置を選択し（ステップ５０）、各演算装置毎にステップ５１～６７の処理を行うことで達成される。ここでは演算装置１を例として扱う。制御記憶障害検出機構１５７により、制御記憶１１５のチェックを行う（ステップ５１、５２）。これは制御記憶１１５に書き込み／読み出し動作を行い、その部分が正常に機能する事を検査するもので、検査する単位は制御記憶のＲＡＭの構成により異なるが、ここでは１ワード単位とする。制御記憶１１５に障害が無い場合、物理アドレス＝論理アドレスとして、マイクロコードテーブル３１０の該当エントリには触らない（ステップ５３）。一方、障害を発見した場合、該当する物理アドレスを持つマイクロプログラムのワードの再配置を行う必要がある。そこ

で、変換テーブル（アドレス対応表）２００の有効フラグ２０１を検索し、空白部を検索する（ステップ５４、５５）。必要な空白部が存在しない場合、コンソール装置１７０にエラーを表示し、マイクロプログラム初期ロードを異常終了させる（ステップ５６、５７）。空白部が存在する場合、アドレス移動するべきワードのマイクロコードテーブル３１０上の該当エントリ内の物理アドレス３４０（演算装置に対応するエントリ）を空白部のアドレスに書き換え（ステップ５８）、変換テーブル２００の該空白部（再配置用エントリ）に該当ワードの論理アドレスを登録し、その有効フラグ２０１を１とすることにより、そのエントリが空白でないことを登録する（ステップ６０）。

【００３６】次に、マイクロコードテーブル３１０の該当エントリのポインタ３１７が指し示すクロスリファレンステーブル３３０の参照ワード論理アドレス３３２、ポインタ３３４を用いて、マイクロコードテーブル３１０の該当エントリのコード内分岐アドレス３５０を書換える（ステップ６１）。これを図５で簡単に説明する。今、制御記憶中のワードａがワードｂに分岐している場合（Ａ）、マイクロコードテーブル３１０のワードｂエントリのポインタ３１７が指すクロスリファレンステーブル内の３３２にはワードａの論理アドレスが登録されている（Ｃ）。ここで、ワードｂの物理アドレスＹ０部に障害が検出されると、ワードｂに物理アドレスの移動（Ｙ０→Ｙ１）が行われるが（Ｂ）、それに伴いワードａの分岐アドレスを書き換えるため、マイクロコードテーブル３１０のワードｂエントリのポインタ３１７が指すクロスリファレンステーブル３３０内の３３２に登録されたワードａの論理アドレスにより、それが指すマイクロコードテーブル３１０のワードａエントリの分岐アドレスを書き換える（Ｃ）。制御記憶の種類によっては分岐アドレスのみならず、パリテイあるいはＥＣＣコードを書き換える必要がある。

【００３７】図４に戻り、マイクロコードテーブル３１０の該当エントリのポインタ３１７が指し示すクロスリファレンステーブル３３０にチェインされている別のエントリが存在するかを調べる（ステップ６２）。エントリが存在すると（例えば図３）、そのワードに分岐する別のワードがあるので、チェインされた全てのエントリわたってステップ６１を繰り返す。

【００３８】以上、ステップ５１～６２をその演算装置の制御記憶全てのワードが尽きるまで繰り返し、当該制御記憶１１５の全てのチェックを行なった後（ステップ６４）、マイクロコードテーブル３１０上の全ワードのオブジェクトコード３１５を、それに物理アドレス３４０を付加してバッファエリア１５３に転送し、そのフラグ６０２を”有効”にする（ステップ６５）。この時、オブジェクトコード内の分岐アドレス３５０及び付加する物理アドレス３４０は、図６の（Ｄ）に示すように、障害発生部位を回避した物理アドレスに書き換えられている。

【００３９】マイクロコードテーブル３１０上の全ワードをバッファエリア１５３に転送後、バッファエリア内のフラグ６０２が”有効”なエントリを、物理アドレス６０１に従い制御記憶１１５にロードする（ステップ６６）。このロード終了後、フラグ６０２を”無効”にする（ステップ６７）。

【００４０】以上、ステップ５０～６７を演算装置毎に繰り返す。こうして、全ての演算装置の制御記憶にマイクロコードがにロードされれば（ステップ６８）、マイクロプログラム初期ロードは終了である。

【００４１】図６は、計算機システム稼働中のアドレス再配置のフローチャートである。以下、図２および図６を参照して稼働時のアドレス再配置の動作を説明する。

【００４２】システム稼働中に制御記憶障害が発生した場合、ＳＶＰ１５０は障害が発生した制御記憶を持つ演算装置を一時停止し、その演算装置に対応する変換テーブル２００を選択する（ステップ６１０）。障害検出機構１５７により制御記憶内の障害発生部のワードを選択する（ステップ６１１）。以下の再配置処理は図４のマイクロプログラム初期ロードとほぼ同じ手順で行う。すなわち、変換テーブル２００を検索して制御記憶の空白部を探し（ステップ６１２）、変換テーブル２００とマイクロコードテーブル３１０内の当該エントリの物理アドレスを書き換え（ステップ６１７、６１８）、更にクロスリファレンステーブル３３０より分岐アドレスの書き換えを行い（ステップ６１９、６２０）、バッファエリア６００への転送（ステップ６２２）、制御記憶への転送を行い（ステップ６２３、６２４）、その後、演算装置の動作を再開するまた、この再配置処理で制御記憶に必要な空白部が存在しなくなった場合、コンソール装置１７０に警告を表示する（ステップ６２５、６２６）。初期ロードと異なるのは、障害発生演算装置と制御記憶の障害発生部位のワードが最初から認識されているため、処理対象が障害発生演算装置かつその制御記憶内の障害発生ワードに限定されることである。

【００４３】次に、図１の計算機システムでのパッチによるマイクロコードの改訂の方法を説明する。アセンブル手段２０にて、マイクロプログラムソースコード２１をアセンブルプログラム及びリンケージプログラム２５によりマイクロプログラムコード２２に変換した後、論理処理部２４により新旧コードの差分情報を作成し、それを（ａ）旧コードからの削除、（ｂ）新規追加、（ｃ）旧コードの変更の順番に抽出する。この差分情報に、論理アドレス、分岐先論理アドレス、及び削除／新規追加／旧コードの変更の種別をフラグとしたデータを付加し、図２の５００に記したフォーマットのパッチデータを作成する。これを転送手段３０を使用して計算機システム１００に転送する。

【００４４】計算機システム１００は受け取ったパッチデータをファイル装置１６０に一時蓄積し、その後、このパッチデータに従いって、マイクロプログラムコード１５５、変換テーブル１５１、１５２を改訂する。これを図２及びと図７のフローチャートを参照して説明する。

【００４５】計算機システム１００は、パッチデータ５００のフラグを参照してデータを削除、新規、変更の順に整列する（ステップ７０１）。データが削除の場合（ステップ７０２）、パッチデータ５００の論理アドレス５０２が示すマイクロコードテーブル３１０中の該当エントリを見つけ、その物理アドレス３１３が示す変換テーブル２００中のエントリ２０２の内容をクリアし、フラグ２０１を”空白”にする（ステップ７０３）。その後、パッチデータ５００の論理アドレス５０２が示すマイクロコードテーブル３１０中の当該エントリの内容３１３、３１４、３１５、３１６、３１７をクリアする（ステップ７０４）。以下、ステップ７０３、７０４を削除データが尽きるまで繰り返す（ステップ７０５）。次に、データが新規の場合（ステップ７０６）、パッチデータ５００の論理アドレス５０２が示すマイクロコードテーブル３１０の該当空きエントリに該パッチデータの内容を入力し（ステップ７０７）、これを新規データ全てを処理するまで繰り返す（ステップ７０８）。また、データが変更の場合（ステップ７１０）、そのワードが分岐先論理アドレスの変更を含むとき分岐アドレス３１４に５０３の内容を入力し（ステップ７１２）、オブジェクトコード３１５に５０４の内容を入力し（ステップ７１３）、これを変更データ全てを処理するまで繰り返す（ステップ７１４）。

【００４６】ここで、分岐アドレスの変更が起きた場合、ワードの相互参照関係が変化するので、クロスリファレンステーブル３３０を再構築する必要がある。これを、図８のフローチャート及び図１０の再構築前と再構築後のテーブルを参照して説明する。

【００４７】最初に、マイクロコードテーブル３１０の全エントリにあるポインタ３１７を初期化する（ステップ８０１）。次に、該マイクロコードテーブル３１０のエントリを指すポインタｉ、クロスリファレンステーブル３３０のエントリを指すポインタｊを先頭に初期設定し（ステップ８０２）、ポインタｉが指すマイクロコードテーブル３１０のエントリの分岐先論理アドレス３１４（図１０ではＢＡ）が示すエントリのポインタ３１７の値（図１０ではX）を、ポインタｊが示すクロスリファレンステーブル３３０のエントリ３３４に入力し、ポインタｉの値を３３２に入力した後、３１７にポインタｊの値を入力する（ステップ８０３）。その後、ポインタｉ、ｊを、テーブル３１０、３３０各々の次のエントリを指すように更新し（ステップ８０４、８０５）、再びステップ８０３を実行する。以降、ステップ８０３－８０５をマイクロコードテーブル３１０のすべてのエントリが終わるまで繰り返す（ステップ８０６）。それが終了した時、図１０の（A）は、同（B）で示す様な各ワード間の前後参照関係ができている。あるワードに分岐するワードが複数個有った場合、クロスリファレンステーブル３３０にはポインタ３３４でチェインされる。

【００４８】以上でパッチの前段階の処理が終わり、実際のパッチロード処理を行なう。これを図９で説明する。

【００４９】計算機システム内の一つの演算装置１を選択する（ステップ９０１）。以下、ステップ９０１－９１８の処理が各々の演算装置毎に繰り返される。まず、マイクロコードテーブル３１０のエントリの中で、先のステップ８０６までの処理で分岐先が変更されたワードを抽出する（ステップ９０２）。これは、ここまでの処理でフラグ３１６にそのことを登録すれば簡単に認識できる。次に、その変更された分岐先の物理アドレスを、マイクロコードテーブル３１０の物理アドレス３４０の該当演算装置番号に対応したエントリから取り出し、自ワードの分岐アドレス３５０のやはり該当演算装置番号に対応したエントリに格納する（ステップ９０３）。分岐先が変更された全てのワードに関して、ステップ９０３の処理を繰り返す（ステップ９０４）。

【００５０】次に、ステップ９０２と同様な手段でパッチで空きエリアに入る新規に作られたワードを抽出する（ステップ９０５）。新たに使用する制御記憶１１５のチェックを行なう為、障害検出機構１５７により制御記憶１１５の該当部分を検査し（ステップ９０６）、障害がなければ、物理アドレス＝論理アドレスとする（ステップ９１１）。障害を発見した場合は、アドレス変換テーブル２００の有効フラグ２０１を検索し、空白部を調べる（ステップ９０７）。必要な空白部が存在しない場合、コンソール装置１７０にエラーを表示し（ステップ９０８）、パッチのロードを異常終了させる（ステップ９０９）。空白部が存在する場合、物理アドレス＝空白部アドレスとする（ステップ９１０）。ここでは物理アドレスと論理アドレスが異る為、このワードに分岐するワードの分岐アドレス３５０を書き換える必要が有る。そのため、その様なワードをクロスレファレンステーブル３３０から検索し、そのワードの分岐アドレス３５０の演算装置に対応したエントリに物理アドレスを入力する（ステップ９１０）。更に、物理アドレスが示す変換テーブル２００のエントリ２０２に元の論理アドレスを登録し、フラグ２０１を１とすことにより、そのエントリが空白でない事を登録し、変換テーブル２００を改定する（ステップ９１２）。ステップ９０６から９１２を、新規に作られたワードが無くなるまで繰り返す（ステップ９１３）。

【００５１】以上の処理が終われば、マイクロコードテーブル３１０から今までの処理で変更の有った全ワード

の物理アドレス３１３、オブジェクトコード３１５に分岐先アドレス３５０をマージしたデータをバッファエリア６００（図１の１５３）に転送する（ステップ９１４）。以下、アドレス再配置の場合と同様に演算装置を一時停止し（ステップ９１５）、バッファエリア６００より制御記憶１１５（又は１１６）にデータをロードする（ステップ９１６）。その後、演算装置を再起動し（ステップ９１７）、当該演算装置へのパッチは終了する。全ての演算装置に関してステップ９０１－９１７を繰り返して（ステップ９１８）、計算機システム１００へのパッチがすべて終了する。

【００５２】次に、固定番地から始まるマイクロプログラムの先頭固定番地が再配置された時の命令起動処理について説明する。図１１は、演算装置内の起動テーブル（図１の１１４、１４４）の働きを説明する図である。起動テーブル１００４の各エントリは演算装置により命令とそれのマイクロプログラムのスタートアドレスが一対一に対応がとられており、命令パイプラインにおいて、命令フェッチ（ステップ１０００、命令デコード（ステップ１００２）と進み、該命令デコード時に演算装置により起動テーブル１００４中の該当エントリが選択され、その先頭アドレスが読み出され（ステップ１００５）、命令実行時に（ステップ１００６）、そのアドレスで制御記憶のマイクロプログラムが起動される（ステップ１００７）。この起動テーブル１００４の先頭アドレスは、ＳＶＰ１５０により書き換えることができる。

【００５３】先のアドレス移動とマイクロプログラムの初期ロード及びパッチにより先頭アドレスが再配置された時（図４のステップ６０、図６のステップ６１８、図９のステップ９１２で起動テーブルに登録されているアドレスが再配置された時）、ＳＶＰ１５０は起動テーブル１００４（図１の１１４、１４４）に新い物理アドレスを書き込む。これによりアドレスが再配置以降のマイクロプログラム起動も動作が保証される。

【００５４】次に、図１の計算機システム１００でのマイクロプログラムトレース採取法を説明する。トレース部１１２、１４２が演算装置１１０、１４０のトレースデータを採取した時、通常、そのデータはＳＶＰ１５０に取り出される。その時、変換テーブル１５１、１５２（図２の２００、２２０）を取り出し、トレースデータと共に保守担当者によりコード転送手段３０を使いマイクロプログラム設計者に届けられる。トレースデータはマイクロプログラムが実行された順の制御記憶１１５、１４６の物理アドレスの順番からなっており、これを変換テーブル２００、２２０を使用して論理アドレスに変換する。あるいは、トレースを採取した時点で、ＳＶＰ１５０が変換テーブル２００、２２０を参照し、トレースデータを論理アドレスに変換してトレースデータを得る事もできる。これにより、マイクロプログラム設計者は、自ら設計したソースコードと整合したトレースデータを得る事ができる。

【００５５】以上説明した実施形態において、マイクロプログラム再配置及びパッチによるマイクロコードの改訂処理では、変換テーブル２００及びマイクロコードテーブル３１０内の演算装置対応の物理アドレス３４０により、再配置による物理アドレス－論理アドレスの対応の変更が、その演算装置に局所化される。これにより、マルチプロセッサシステムの場合、アドレス再配置は障害を起こした演算装置以外には波及せず、それら演算装置の制御記憶には無効領域が生じない。

【００５６】又、図４のステップ５５－５７、図６のステップ６１３－６１４、図９のステップ９０７－９０９をみればわかる様に、制御記憶の空き領域が十分でない、即ち、アドレス移動、マイクロプログラムの初期ロード、又はパッチに必要な量が存在しない場合、その事を自動的に警告メッセージとして表示している。これにより、保守操作員はシステムダウンを予想し、計算機システム保守の契機を知ることができる。

【００５７】

【発明の効果】本発明の制御記憶の障害回避方法によれば、以下のような効果が得られる。

（１）顧客サイト毎に存在する計算機システムに対して、論理アドレスをキーに共通のマイクロプログラムコード及びパッチを適用することができ、マイクロプログラムの保守操作性が向上する。

（２）マイクロプログラム再配置時に、移動先の物理アドレスを、演算装置に設けられた起動アドレスを登録したテーブルのエントリに上書きすることにより、アドレスの再配置が行われたマイクロプログラムを持つ演算装置の実行を保証できる。

【００５８】（３）計算機システム内演算装置の実行状況をトレースする際、トレース情報の物理アドレスを論理アドレスに変換して出力することにより、該出力されたトレース情報と論理アドレスをキーとしたマイクロプログラムのソースリストとのアドレス整合性を確保することができ、任意の計算機システムで採取されたトレース情報を共通のソースリストにより解析できる。

【図面の簡単な説明】

【図１】本発明における計算機システムの制御記憶のアドレス再配置を行なう全体的機構を示した図である。

【図２】図１のＳＶＰ上のアドレス再配置に関するテーブル類の詳細構成例を示した図である。

【図３】テーブル上のマイクロプログラムワード間の相互関係を示した図である。

【図４】本発明における計算機システムの制御記憶のマイクロプログラム初期ロードを示したフローチャートである。

【図５】制御記憶のアドレス再配置を行なう時の制御記憶とテーブルの取扱いを示した図である。

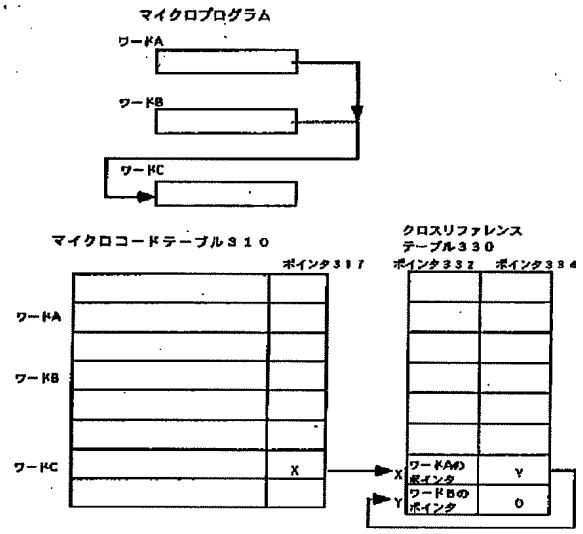
15

【図６】制御記憶のアドレス再配置のフローチャートである。
【図７】計算機システムにパッチを適用する時のフローチャートの一部である。
【図８】計算機システムにパッチを適用する時のフローチャートの続きである。
【図９】計算機システムにパッチを適用する時のフローチャートの更に続きである。
【図１０】クロスリファレンステーブルの再構築を説明する図である。
【図１１】起動テーブルの働きを示した図である。
【符号の説明】
１０　マイクロプログラム変更手段
２０　アセンブル（リンケージ）手段
２１　マイクロプログラムソースコード
２２　マイクロプログラムコード
２３　マイクロプログラムパッチデータ
２４　新旧マイクロプログラムソースコードよりパッチデータを生成する手段

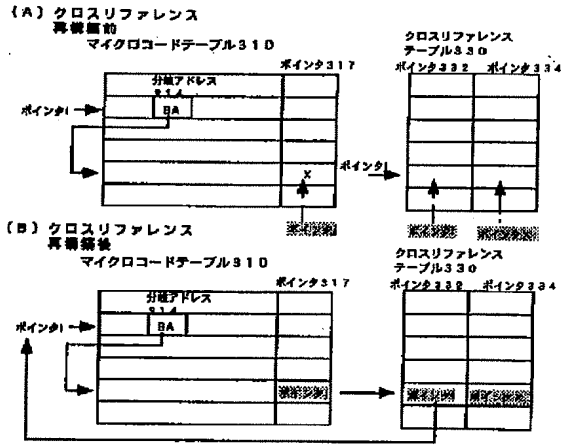
16

＊２５　アセンブル&リンケージプログラム
３０　マイクロプログラムコード&パッチデータ転送手段
１００　計算機システム
１１０、１４０　計算機システム内の演算装置
１１２、１４２　マイクロプログラムトレース採取部
１１４、１４４　マイクロプログラム起動テーブル
１１５、１４６　制御記憶
１２０　主記憶装置
１２５　主記憶装置のハードウエア専用領域（ＨＳＡ）
１４２　マイクロプログラムトレース採取装置
１５０　ＳＶＰ
１５１、１５２　アドレス変換テーブル
１５３　マイクロプログラム改定用のバッファエリア
１５５　ＳＶＰ１５０上のマイクロプログラムコード
１５７　制御記憶障害検出機構
１６０　ファイル装置
１７０　コンソール装置

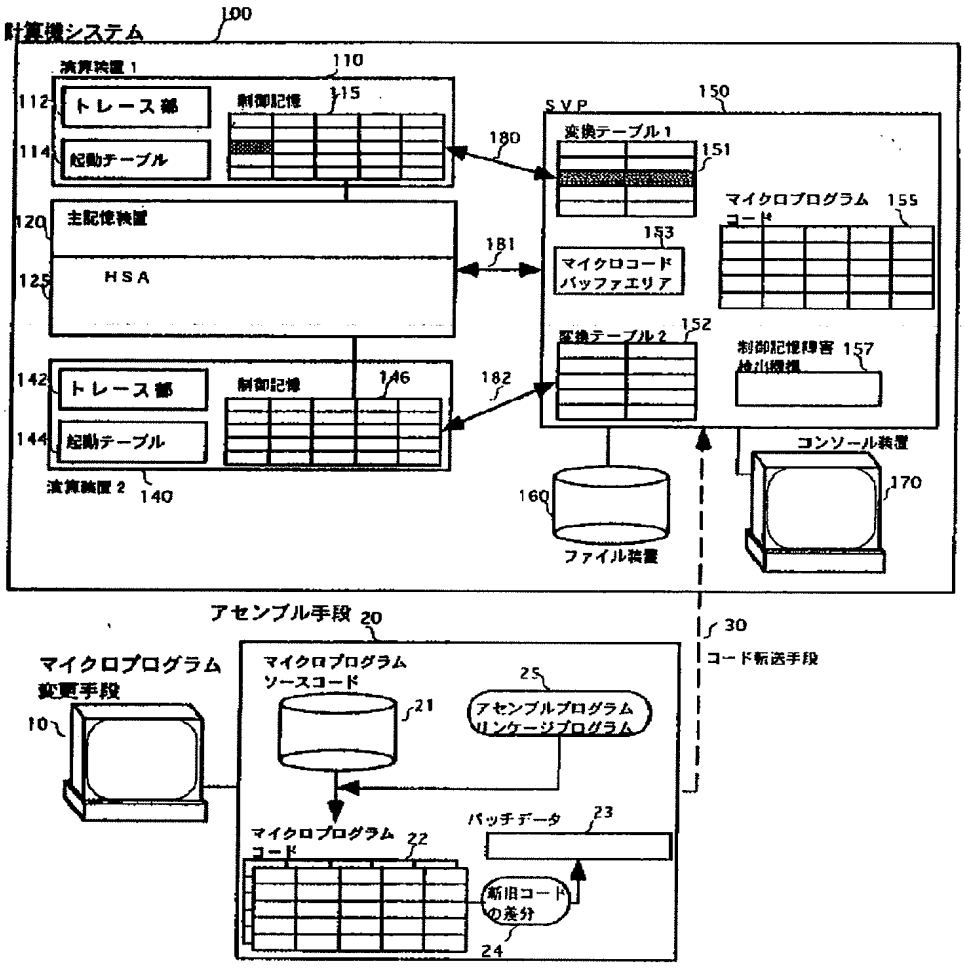
【図３】



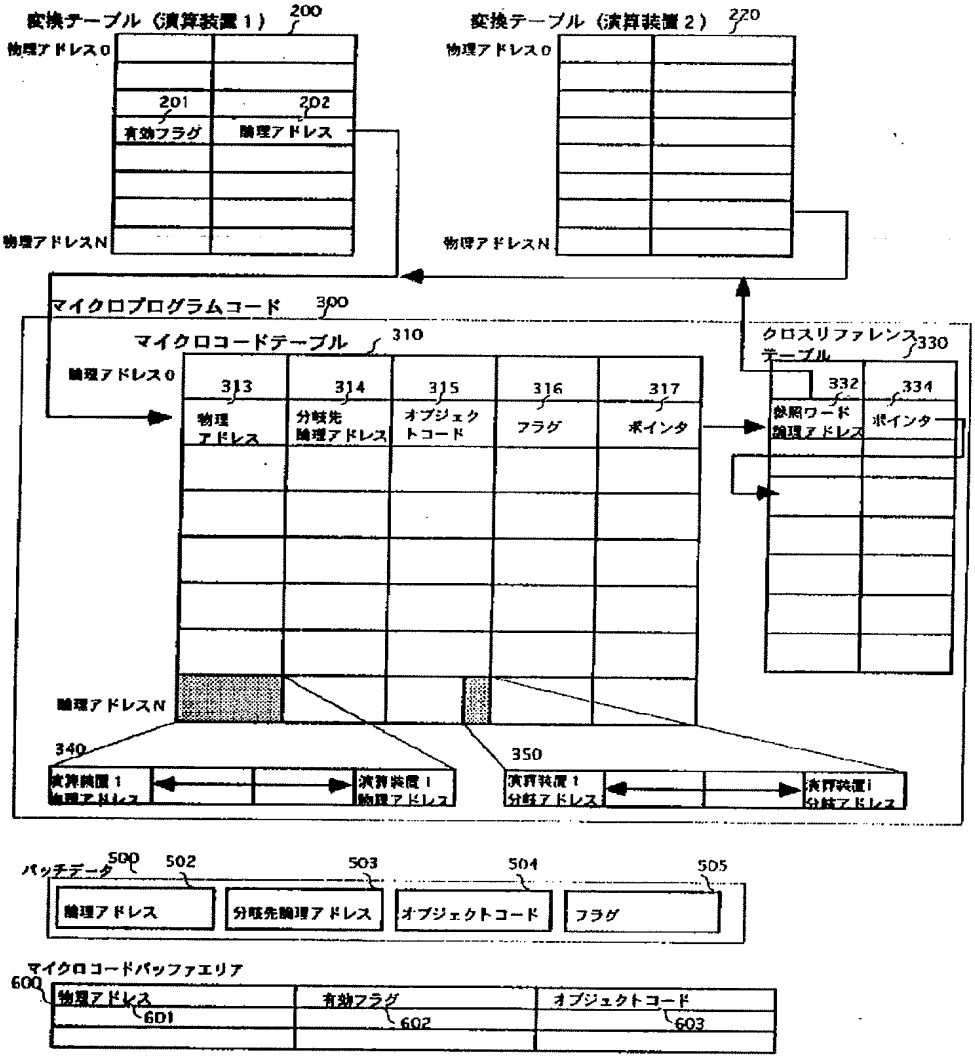
【図１０】



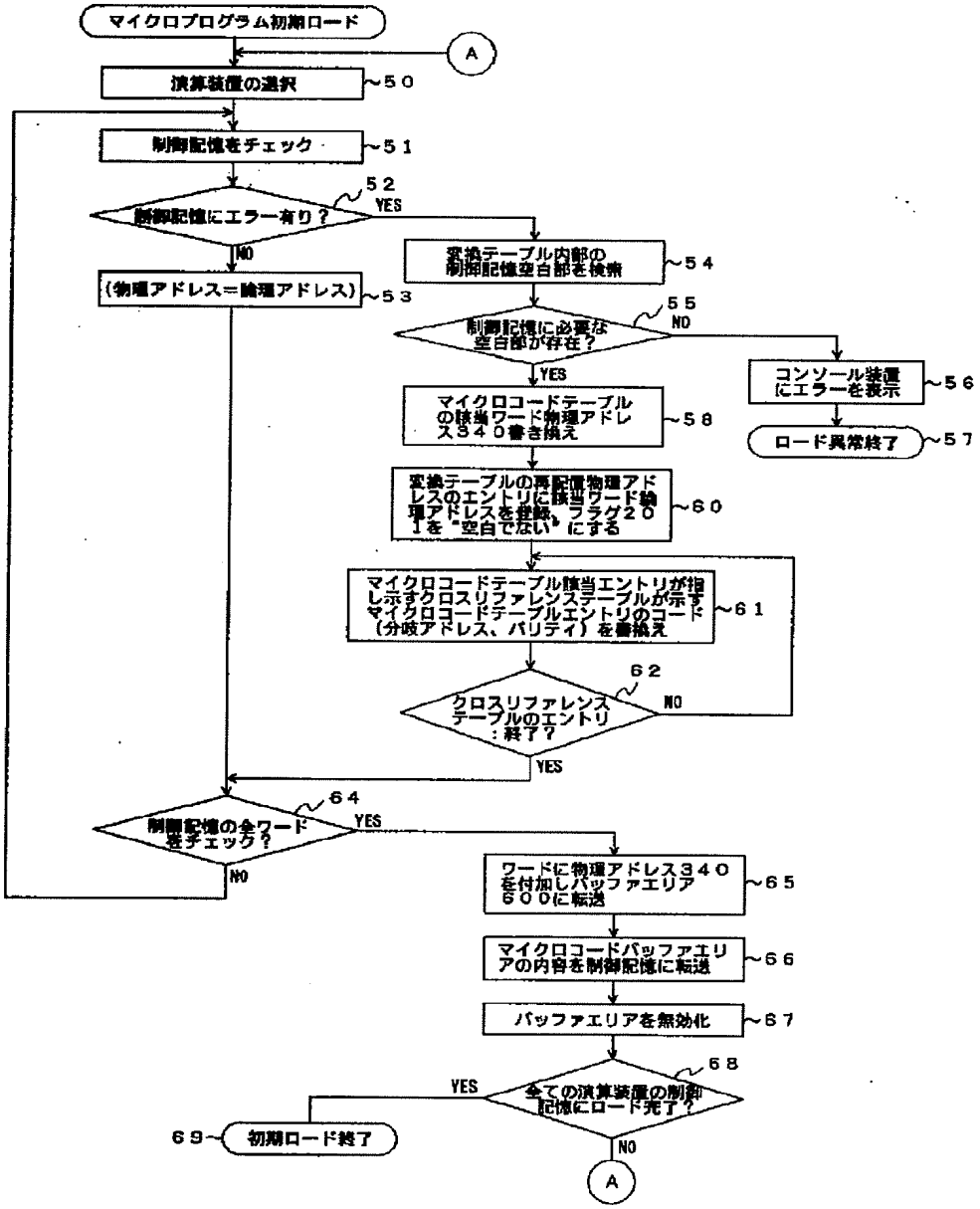
【図１】


計算機システム
100
演算装置１
110
トレース部
112
制御記憶
115
起動テーブル
114
主記憶装置
120
HSA
125
トレース部
142
制御記憶
146
起動テーブル
144
演算装置２
140
SVP
150
変換テーブル１
151
180
マイクロプログラム
コード
155
153
マイクロコード
バッファエリア
181
変換テーブル２
152
制御記憶障害
検出機構
157
182
コンソール装置
170
160
ファイル装置
アセンブル手段
20
30
コード転送手段
マイクロプログラム
変更手段
10
マイクロプログラム
ソースコード
21
25
アセンブルプログラム
リンケージプログラム
パッチデータ
23
マイクロプログラム
コード
22
新旧コード
の差分
24

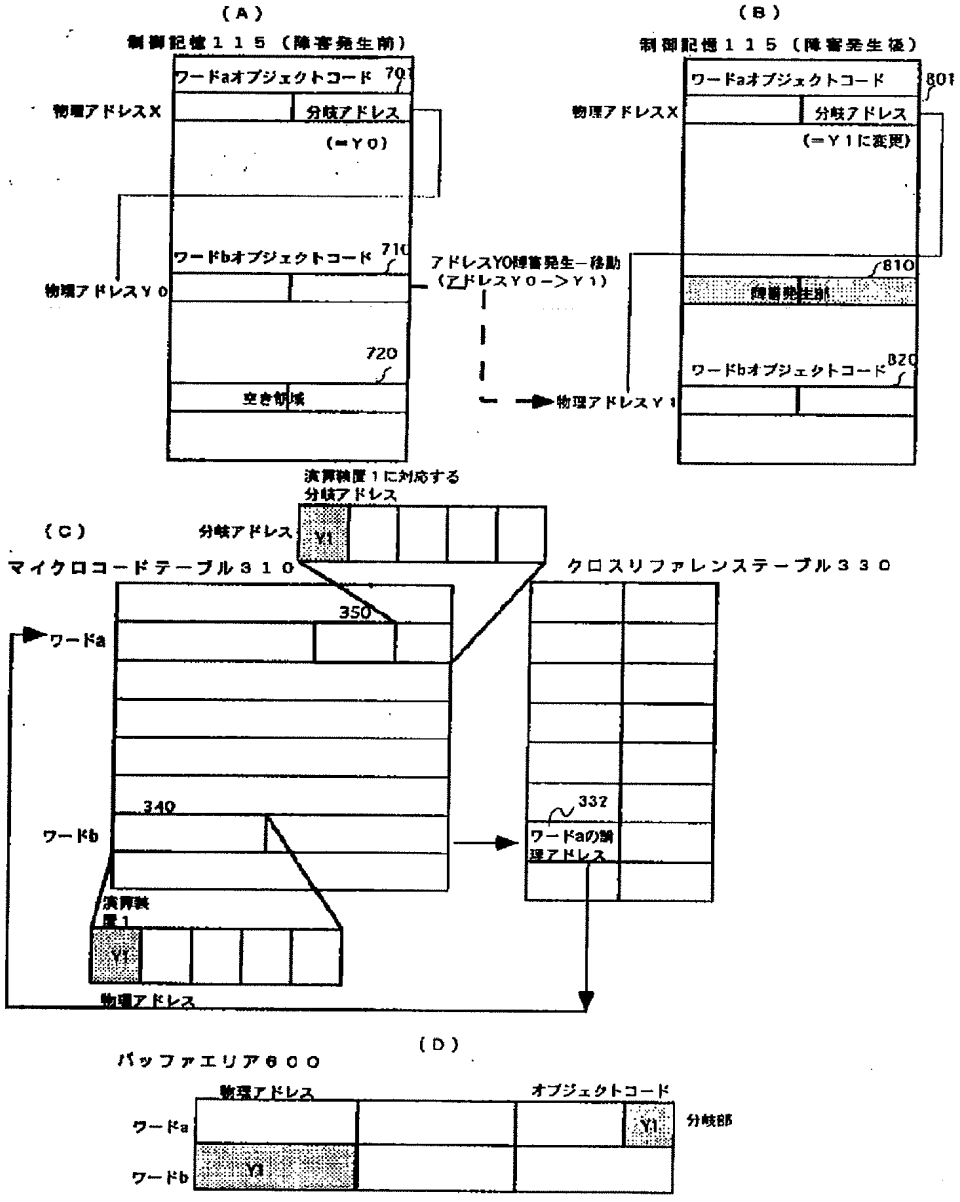
【図２】


変換テーブル（演算装置１）
200
物理アドレス０
201
有効フラグ
202
論理アドレス
物理アドレスN
変換テーブル（演算装置２）
220
物理アドレス０
物理アドレスN
マイクロプログラムコード
300
マイクロコードテーブル
310
論理アドレス０
313
物理アドレス
314
分岐先論理アドレス
315
オブジェクトコード
316
フラグ
317
ポインタ
クロスリファレンステーブル
330
332
参照ワード論理アドレス
334
ポインター
論理アドレスN
340
演算装置１物理アドレス
演算装置ｉ物理アドレス
350
演算装置１分岐アドレス
演算装置ｉ分岐アドレス
パッチデータ
500
502
論理アドレス
503
分岐先論理アドレス
504
オブジェクトコード
505
フラグ
マイクロコードバッファエリア
600
物理アドレス
601
有効フラグ
602
オブジェクトコード
603

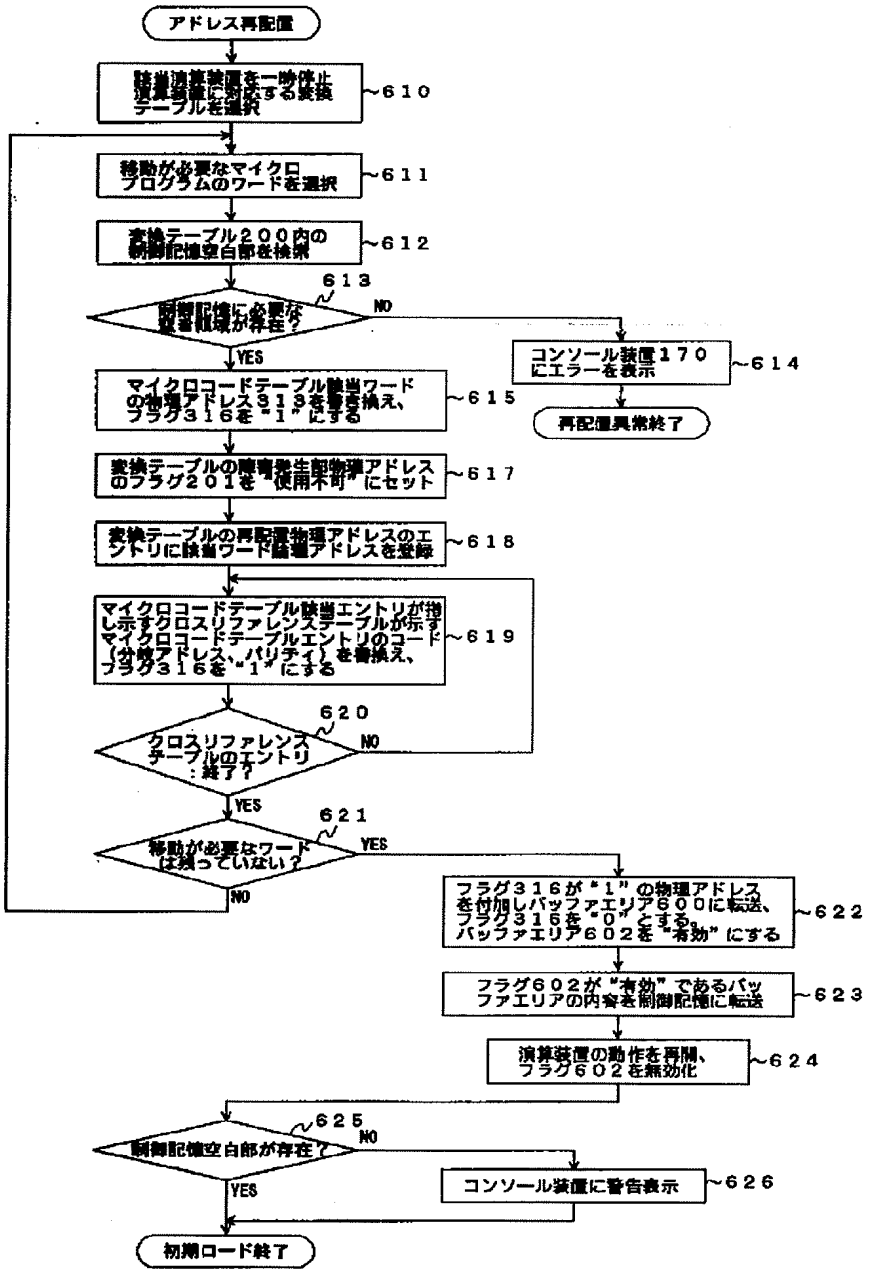
【図４】


マイクロプログラム初期ロード
A
演算装置の選択
50
制御記憶をチェック
51
52
制御記憶にエラー有り？
YES
NO
(物理アドレス＝論理アドレス)
53
変換テーブル内部の制御記憶空白部を検索
54
55
制御記憶に必要な空白部が存在？
NO
YES
コンソール装置にエラーを表示
56
ロード異常終了
57
マイクロコードテーブルの該当ワード物理アドレス３４０書き換え
58
変換テーブルの再配置物理アドレスのエントリに該当ワード論理アドレスを登録、フラグ２０１を"空白でない"にする
60
マイクロコードテーブル該当エントリが指し示すクロスリファレンステーブルが示すマイクロコードテーブルエントリのコード（分岐アドレス、パリティ）を書換え
61
62
クロスリファレンステーブルのエントリ：終了？
NO
YES
64
制御記憶の全ワードをチェック？
YES
NO
ワードに物理アドレス３４０を付加しバッファエリア６００に転送
65
マイクロコードバッファエリアの内容を制御記憶に転送
66
バッファエリアを無効化
67
68
全ての演算装置の制御記憶にロード完了？
YES
NO
69
初期ロード終了
A

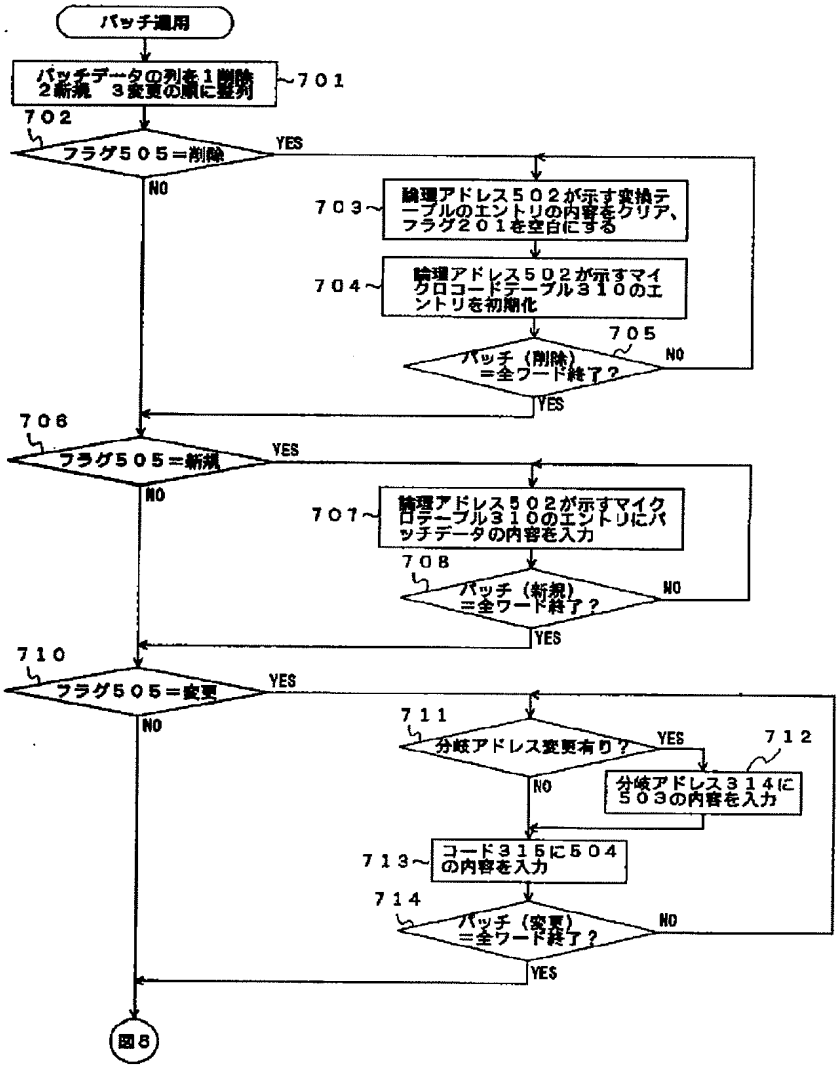
【図５】


（Ａ）
制御記憶１１５（障害発生前）
ワードaオブジェクトコード
701
物理アドレスX
分岐アドレス
（＝Y0）
ワードbオブジェクトコード
710
物理アドレスY0
720
空き領域
アドレスY0障害発生→移動
（アドレスY0→Y1）
（Ｂ）
制御記憶１１５（障害発生後）
ワードaオブジェクトコード
801
物理アドレスX
分岐アドレス
（＝Y1に変更）
810
障害発生!!
ワードbオブジェクトコード
820
物理アドレスY1
（Ｃ）
演算装置１に対応する
分岐アドレス
分岐アドレス
Y1
マイクロコードテーブル３１０
クロスリファレンステーブル３３０
350
ワードa
340
ワードb
332
ワードaの論理アドレス
演算装置１
Y1
物理アドレス
（Ｄ）
バッファエリア６００
物理アドレス
オブジェクトコード
ワードa
Y1
分岐部
ワードb
Y1

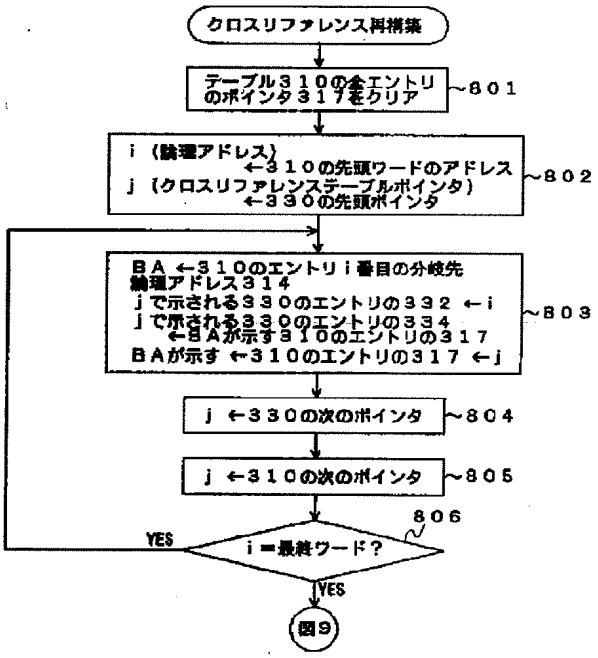
【図６】


アドレス再配置
該当演算装置を一時休止
演算装置に対応する変換
テーブルを選択
610
移動が必要なマイクロ
プログラムのワードを選択
611
変換テーブル２００内の
制御記憶空白部を検索
612
613
制御記憶に必要な
空き領域が存在？
NO
YES
コンソール装置１７０
にエラーを表示
614
再配置異常終了
マイクロコードテーブル該当ワード
の物理アドレス３１３を書き換え、
フラグ３１６を“１”にする
615
変換テーブルの障害発生部物理アドレス
のフラグ２０１を“使用不可”にセット
617
変換テーブルの再配置物理アドレスのエ
ントリに該当ワード論理アドレスを登録
618
マイクロコードテーブル該当エントリが指
し示すクロスリファレンステーブルが示す
マイクロコードテーブルエントリのコード
（分岐アドレス、パリティ）を書換え、
フラグ３１６を“１”にする
619
620
クロスリファレンス
テーブルのエントリ
：終了？
NO
YES
621
移動が必要なワード
は残っていない？
YES
NO
フラグ３１６が“１”の物理アドレス
を付加しバッファエリア６００に転送、
フラグ３１６を“０”とする、
バッファエリア６０２を“有効”にする
622
フラグ６０２が“有効”であるバッ
ファエリアの内容を制御記憶に転送
623
演算装置の動作を再開、
フラグ６０２を無効化
624
625
制御記憶空白部が存在？
NO
YES
コンソール装置に警告表示
626
初期ロード終了

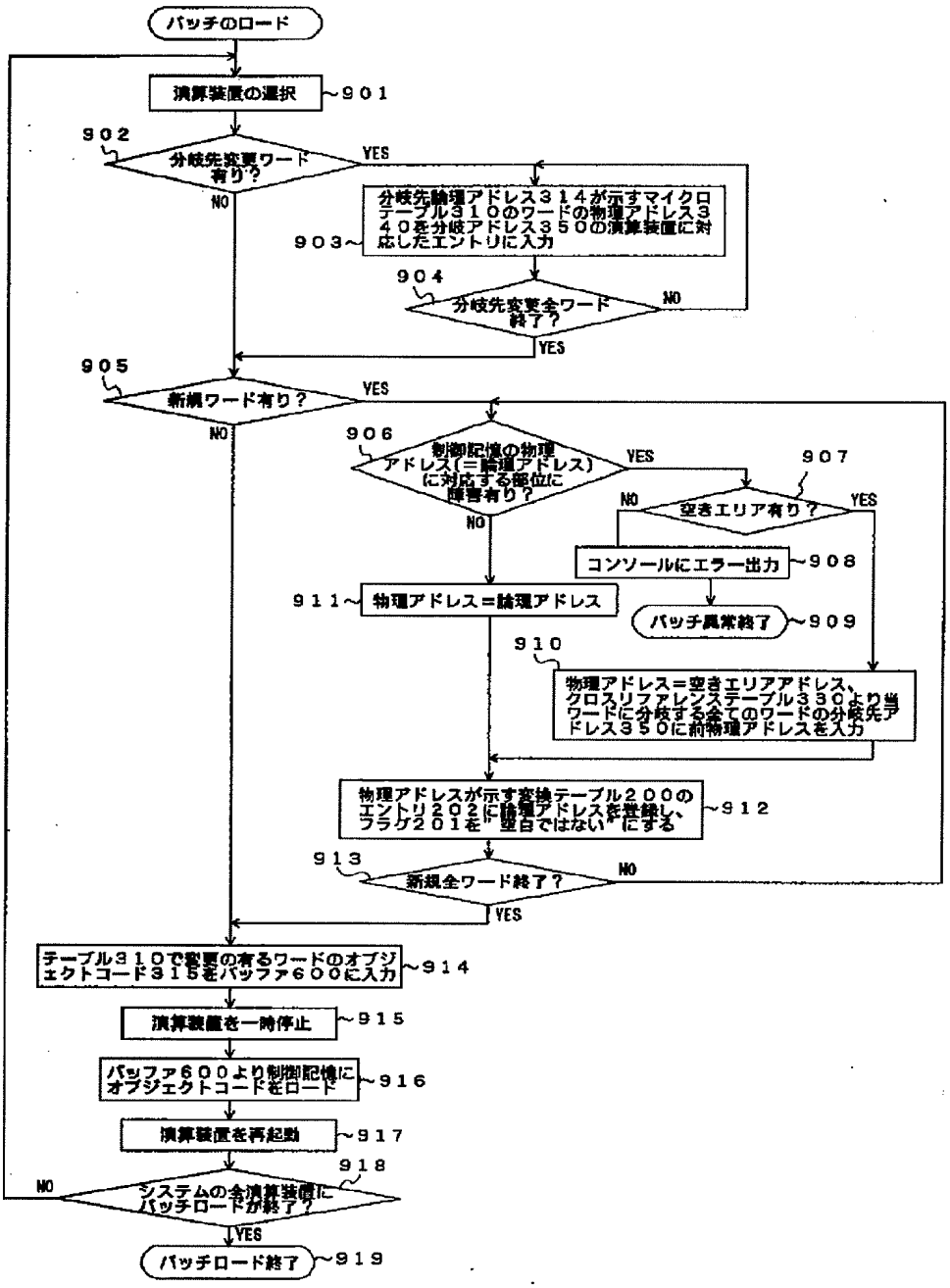
【図７】


パッチ適用
パッチデータの列を１削除
２新規　３変更の順に整列
701
702
フラグ５０５＝削除
YES
NO
703
論理アドレス５０２が示す変換テーブルのエントリの内容をクリア、フラグ２０１を空白にする
704
論理アドレス５０２が示すマイクロコードテーブル３１０のエントリを初期化
705
パッチ（削除）＝全ワード終了？
NO
YES
706
フラグ５０５＝新規
YES
NO
707
論理アドレス５０２が示すマイクロテーブル３１０のエントリにパッチデータの内容を入力
708
パッチ（新規）＝全ワード終了？
NO
YES
710
フラグ５０５＝変更
YES
NO
711
分岐アドレス変更有り？
YES
712
分岐アドレス３１４に５０３の内容を入力
NO
713
コード３１５に５０４の内容を入力
714
パッチ（変更）＝全ワード終了？
NO
YES
図８

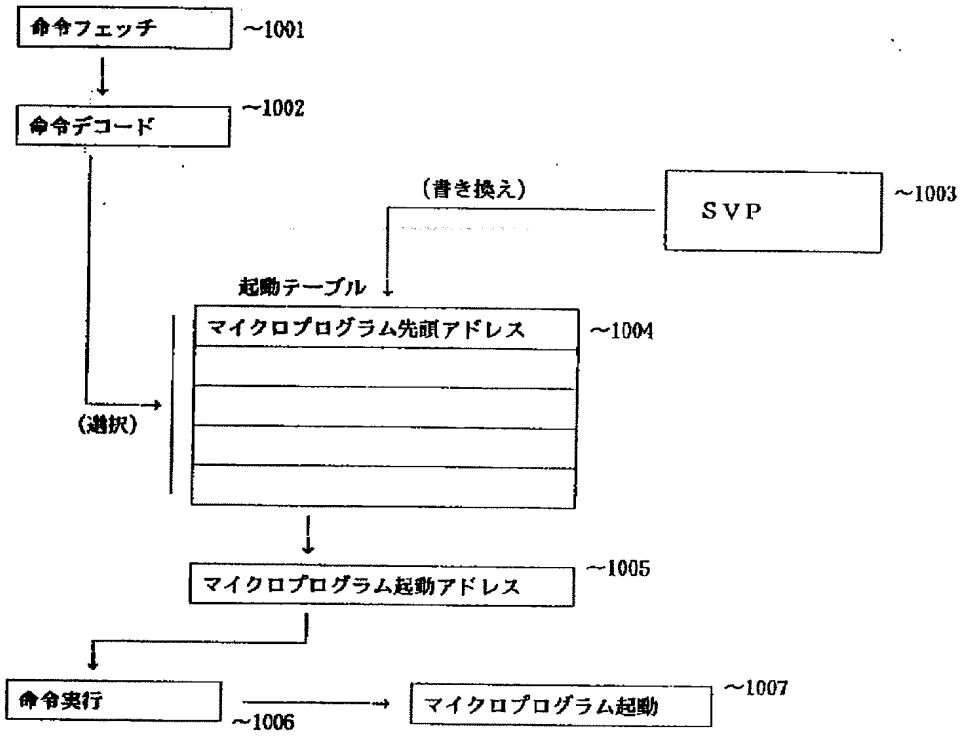
【図８】


クロスリファレンス再構築
テーブル３１０の全エントリのポインタ３１７をクリア
801
i（論理アドレス）←３１０の先頭ワードのアドレス
j（クロスリファレンステーブルポインタ）←３３０の先頭ポインタ
802
ＢＡ←３１０のエントリｉ番目の分岐先論理アドレス３１４
ｊで示される３３０のエントリの３３２←ｉ
ｊで示される３３０のエントリの３３４←ＢＡが示す３１０のエントリの３１７
ＢＡが示す←３１０のエントリの３１７←ｊ
803
ｊ←３３０の次のポインタ
804
ｊ←３１０の次のポインタ
805
806
ｉ＝最終ワード？
YES
YES
図９

【図９】


パッチのロード
演算装置の選択
９０１
９０２
分岐先変更ワード有り？
YES
NO
９０３
分岐先論理アドレス３１４が示すマイクロテーブル３１０のワードの物理アドレス３４０を分岐アドレス３５０の演算装置に対応したエントリに入力
９０４
分岐先変更全ワード終了？
NO
YES
９０５
新規ワード有り？
YES
NO
９０６
制御記憶の物理アドレス（＝論理アドレス）に対応する部位に障害有り？
YES
NO
９０７
空きエリア有り？
NO
YES
コンソールにエラー出力
９０８
９１１
物理アドレス＝論理アドレス
パッチ異常終了
９０９
９１０
物理アドレス＝空きエリアアドレス、クロスリファレンステーブル３３０より当ワードに分岐する全てのワードの分岐先アドレス３５０に前物理アドレスを入力
物理アドレスが示す変換テーブル２００のエントリ２０２に論理アドレスを登録し、フラグ２０１を"空白ではない"にする
９１２
９１３
新規全ワード終了？
NO
YES
テーブル３１０で変更の有るワードのオブジェクトコード３１５をバッファ６００に入力
９１４
演算装置を一時停止
９１５
バッファ６００より制御記憶にオブジェクトコードをロード
９１６
演算装置を再起動
９１７
９１８
システムの全演算装置にパッチロードが終了？
NO
YES
パッチロード終了
９１９

【図１１】


命令フェッチ
～1001
命令デコード
～1002
（書き換え）
ＳＶＰ
～1003
起動テーブル
マイクロプログラム先頭アドレス
～1004
（選択）
マイクロプログラム起動アドレス
～1005
命令実行
～1006
マイクロプログラム起動
～1007